UNIT – 5 Android Widgets (UI)

1. What is Android Widgets?

A widget is a small gadget or control of your android application placed on the home screen. Widgets can be very handy as they allow you to put your favourite applications on your home screen in order to quickly access them. You have probably seen some common widgets, such as music widget, weather widget, clock widget e.t.c

Widgets could be of many types such as information widgets, collection widgets, control widgets and hybrid widgets. Android provides us a complete framework to develop our own widgets.

2. How to hide Title bar in Android App Development?

In this example, we are going to explain how to hide the title bar.

The requestWindowFeature(Window.FEATURE_NO_TITLE) method of Activity must be called to hide the title. But, it must be coded before the setContentView method.

Code that hides title bar of activity

The getSupportActionBar() method is used to retrieve the instance of ActionBar class. Calling the hide() method of ActionBar class hides the title bar.

- requestWindowFeature(Window.FEATURE_NO_TITLE);//will hide the title
- getSupportActionBar().hide(); //hide the title bar

Full Example that hides title bar

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="first.javatpoint.com.hidetitlebar.MainActivity">
```

```
<TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</android.support.constraint.ConstraintLayout>
```

MainActivity.java

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE); //will hide the title
        getSupportActionBar().hide(); // hide the title bar
setContentView(R.layout.activity_main);

    }
}
```

3. Explain Screen Orientation in Android Studio.

The **screenOrientation** is the attribute of activity element. The orientation of android activity can be portrait, landscape, sensor, unspecified etc. You need to define it in the AndroidManifest.xml file.

Syntax:

```
<activity android:name="package_name.Your_ActivityName"
    android:screenOrientation="orirntation_type">
</activity>
```

Example:

```
<activity android:name=" example.javatpoint.com.screenorientation.MainActivity"
    android:screenOrientation="portrait">
</activity>
<activity android:name=".SecondActivity"
    android:screenOrientation="landscape">
</activity>
```

The common values for screenOrientation attribute are as follows:

Value	Description
unspecified	It is the default value. In such case, system chooses the orientation.
portrait	taller not wider
landscape	wider not taller
sensor	orientation is determined by the device orientation sensor.

4. Explain Form Widget Palettes (Controls) in Android Studio.

MOBILE APPLICATION DEVELOPMENT - 1

The Android Studio palette contains various different views that we can drag onto the "design editor" representing the display of an Android device. These views are divided into categories and are all covered in the following sections.

1. TextView :

Android Textview is user interface element that shows texts to the user and optionally allow them to edit it. However, Text editing is disabled by default. We need to configure the basic class by adding some attributes to make it editable.

Different Attributes of Android TextView Widget

Below are the different attributes of android Textview widget that are commonly used to customise the Textview. However, you can check the android official documentation website for complete list of Textview attributes. Here, we are listing some of the attributes that we will need commonly.

Sr.	XML Attributes	Description
1	android:autoLink	Use this attribute when you want to automatically detect urls or emails and show it as clickable link.
2	android:autoText	If it is set, it means Textview has a textual input method and automatically corrects some common spelling errors.
3	android:capitalize	If it is set, it means Textview has a textual input method and should automatically capitalize what the user types.
4	android:id	This is unique id of the widget to uniquely identify the widget.
5	android:cursorVisible	This is used to make cursor visible or invisible. Default is false.
6	android:drawableBottom	Use this attribute when you want to show any drawable(images etc.) below the text.

7	android:drawableEnd	Use this attribute when you want to show any drawable(images etc.) to end of the text.
8	android:drawableLeft	Use this attribute when you want to show any drawable(images etc.) to left of the text.
9	android:drawablePadding	Use this attribute when you want to add padding to the drawable(images etc.).
10	android:drawableRight	Use this attribute when you want to show any drawable(images etc.) to right of the text.
11	android:drawableStart	Use this attribute when you want to show any drawable(images etc.) to start of the text.
12	android:drawableTop	Use this attribute when you want to show any drawable(images etc.) to top of the text.
13	android:ellipsize	This attribute causes the text, longer than the view, to be ellipsized at the end.
14	android:ems	Makes the Textview be exactly this many ems wide.
15	android:gravity	This is used to align the text (by x-axis, y-axis or both) within this Textview.
16	android:height	This is used to provide the height to the Textview.
17	android:hint	Hint to be shown when there is no text in the Textview.
18	android:inputType	This is used to define what are the types of data can be entered by the user for this Textview. For example, Phone, Password, Number, Date, Time etc.
19	android:lines	If you want to set height of the Textview by number of lines, you can do it using this attribute. For example, android:lines=”2”, it means height of Textview will be 2 lines.
20	android:maxHeight	Use to set the maximum height of the Textview.

21	android:minHeight	Use to set the minimum height of the Textview.
22	android:maxLength	Use to set the maximum character length of the Textview.
23	android:maxLines	Use to set the maximum lines Textview can have.
24	android:minLines	Use to set the minimum lines Textview can have.
25	android:maxWidth	Use to set the maximum width Textview can have.
26	android:minWidth	Use to set the minimum width Textview can have
27	android:text	Use to set the text of the Textview
28	android:textAllCaps	Use this attribute when you want to show text in capital letters. It takes true or false value only. Default is false.
29	android:textColor	Use to set the color of the text.
30	android:textSize	Use to set the size of the text.
31	android:textStyle	Use to set the style of the text. For example, bold, italic, bolditalic.
32	android:typeface	Use to set typeface of the text. For example, normal, sans, serif, monospace.
33	android:width	Use to set width of the TextView.

2. EditText :

An Android EditText widget is user interface that are used to take input from user and modify the text. While defining EditText widget, we can also specify **android:inputType** attribute. Based on the value provided in this attribute, keyboard type shown also changes acceptable characters and appearance of the edit text.

Different attributes of Android EditText Widget

Different attributes that are used to customise the EditText are listed below. However, you can check android official documentation for EditText to see the complete list of it's attributes. Here, we are listing the commonly used attributes.

Attributes in EditText are inherited from TextView and View. Some of the popular attributes are –

Sr.	XML Attributes	Description
1	android:background	Sets background to this View.
2	android:backgroundTint	Sets tint to the background.
3	android:clickable	Set true when you want to make this View clickable. Otherwise, set false.
4	android:drawableBottom	Sets drawable to bottom of the text.
5	android:drawableEnd	Sets drawable to end of the text.
6	android:drawableLeft	Sets drawable to left of the text.
7	android:drawablePadding	Sets padding to drawable.
8	android:drawableRight	Sets drawable to right of the text.
9	android:drawableStart	Sets drawable to start of the text.
10	android:drawableTop	Sets drawable to top of the text.
11	android:elevation	Sets elevation to this view.
12	android:gravity	Sets gravity of the text. For example, center, horizontal_center, vertical_center etc.
13	android:height	Sets height of the EditText.
14	android:hint	Hint to be shown when there is no text in the EditText.

15	android:inputMethod	It sets, it specifies that edittext should use specified input method.
16	android:inputType	This is used to define what are the types of data that can be entered by the user for this View. For example, Phone, Password, Number, Date, Time etc. Characters acceptable through keyboard will also change accordingly.
17	android:lines	If you want to set height of the View by number of lines, you can do it using this attribute. For example, android:lines=”2”, it means height of View will be 2 lines.
18	android:maxHeight	Sets maximum height of the View.
19	android:minHeight	Sets minimum height of the View.
20	android:maxLength	Sets maximum character length that can be entered in the View.
21	android:maxLines	Sets maximum lines this View can have.
22	android:minLines	Sets minimum lines this View can have.
23	android:maxWidth	Sets maximum width this View can have.
24	android:minWidth	Sets minimum width this View can have.
25	android:numeric	If sets, it specifies that EditText has numeric input method.
26	android:password	Use this attribute if you want to show the entered text as password dots. For example, If you enter ABCD, it will be shown as ****.
27	android:phoneNumber	If set, specifies that this EditText has a phone number input method.
28	android:text	Sets the text of the EditText

29	android:textAllCaps	Use this attribute to show the text in capital letters.
30	android:textColor	Sets color of the text.
31	android:textSize	Sets size of the text.
32	android:textStyle	Sets style of the text. For example, bold, italic, bolditalic etc.
33	android:typeface	Sets typeface of the text. For example, normal, sans, serif, monospace.
34	android:width	Sets width of the TextView.

3. Button

Android Button is user interface that user can tap or click to perform some action.

Different Attributes Of Android Button Widget

Different attributes that are used in button are list below. However, if you want to see the complete list of attributes, you need to visit android official documentation on Button widget. Here, we are listing commonly used attributes.

Attributes in Button are inherited from TextView and View. Some of the popular attributes are –

Sr.	XML Attributes	Description
1	android:background	Sets background to this View.
2	android:backgroundTint	Sets tint to the background.
3	android:clickable	Sets true when you want to make this View clickable. Otherwise, set false.

4	android:drawableBottom	This is drawable to be drawn at bottom of the text.
5	android:drawableEnd	This is drawable to be drawn to end of the text.
6	android:drawableLeft	This is drawable to be drawn to left of the text.
7	android:drawablePadding	This is padding of the drawable.
8	android:drawableRight	This is drawable to be drawn to right of the text.
9	android:drawableStart	This is drawable to be drawn to start of the text.
10	android:drawableTop	This is drawable to be drawn at top of the text.
11	android:text	Sets the text of the EditText.
12	android:textAllCaps	Shows text in capital letters.
13	android:textColor	Sets color of the text.
14	android:textSize	Sets size of the text.
15	android:textStyle	Sets style of the text. For example, bold, italic, bolditalic etc.
16	android:typeface	Sets typeface of the text. For example, normal, sans, serif, monospace.

4. Toggle Button

Android Toggle button is a widget that are used to display checked/unchecked state as button with light indicator and by default accompanied with the text `ON` or `OFF`.

It is subclass of Compound button.

Different Attributes of Android Toggle Button Widget

Attributes in toggle button are –

Sr.	XML Attributes	Description
1	android:disabledAlpha	This is value of alpha you want to set when indicator is disabled.
2	android:textOff	Text to be shown when toggle button is in OFF state.
3	android:textOn	Text to be shown when toggle button is in ON state.

Attributes in Toggle Button are also inherited from TextView and Compound Button. Some of the popular attributes inherited from TextView are –

Sr.	XML Attributes	Description
1	android:background	Sets background to this View.
2	android:backgroundTint	Sets tint to the background.
3	android:clickable	Sets true when you want to make this View clickable. Otherwise, set false.
4	android:drawableBottom	This is drawable to be drawn at bottom of the text.
5	android:drawableEnd	This is drawable to be drawn to end of the text.
6	android:drawableLeft	This is drawable to be drawn to left of the text.
7	android:drawablePadding	This is padding of the drawable.
8	android:drawableRight	This is drawable to be drawn to right of the text.

9	android:drawableStart	This is drawable to be drawn to start of the text.
10	android:drawableTop	This is drawable to be drawn at top of the text.
11	android:text	Sets the text of the TextView.
12	android:textAllCaps	Shows text in capital letters.
13	android:textColor	Sets color of the text.
14	android:textSize	Sets size of the text.
15	android:textStyle	Sets style of the text. For example, bold, italic, bolditalic etc.
16	android:typeface	Sets typeface of the text. For example, normal, sans, monospace etc.

Attributes in Toggle Button inherited from Compound Button are –

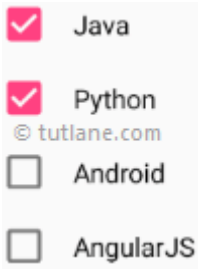
Sr.	XML Attributes	Description
1	android:button	Drawable used for the button graphic (for example, checkbox and radio button).
2	android:buttonTint	Tint to apply to the button graphic.

5. Checkbox:

In android, CheckBox is a two-states button that can be either checked (ON) or unchecked (OFF) and it will allow users to toggle between the two states (ON / OFF) based on the requirements.

Generally, we can use multiple CheckBox controls in android application to allow users to select one or more options from the set of values.

Following is the pictorial representation of using CheckBox control in android applications.



By default, the android CheckBox will be in the OFF (Unchecked) state. We can change the default state of CheckBox by using android:checked attribute.

In case, if we want to change the state of CheckBox to ON (Checked), then we need to set android:checked = "true" in our XML layout file.

In android, we can create CheckBox control in two ways either in the XML layout file or create it in the Activity file programmatically.

Create CheckBox in XML Layout File

Following is the sample way to define CheckBox control in XML layout file in android application.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent" android:layout_height="match_parent">
<CheckBox
   android:id="@+id/chk1"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:checked="true"
   android:text="Java" /> </RelativeLayout>
```

If you observe above code snippet, here we defined CheckBox control and setting CheckBox state ON using android:checked attribute in xml layout file.

Handle Android CheckBox Click Events

Generally, whenever the user clicks on CheckBox to Select or Deselect the CheckBox object will receive an on-click event.

Define CheckBox Click Event in Activity File
In android, we can define CheckBox click event programmatically in Activity file rather than XML layout file.

To define checkbox click event programmatically, create View.OnClickListener object and assign it to the button by calling setOnClickListener(View.OnClickListener) like as shown below.

```
CheckBox chk = (CheckBox) findViewById(R.id.chk1);
chk.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      boolean checked = ((CheckBox) v).isChecked();
      // Check which checkbox was clicked
      if (checked){
         // Do your coding
      }
      else{
         // Do your coding
      }
   }
});
```

This is how we can handle CheckBox click events in android applications based on our requirements.

Android CheckBox Control Attributes
The following are some of the commonly used attributes related to CheckBox control in android applications.

Attribute	Description
android:id	It is used to uniquely identify the control
android:checked	It is used to specify the current state of checkbox

Attribute	Description
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text for a checkbox.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for checkbox control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:onClick	It's the name of the method to invoke when the checkbox clicked.
android:visibility	It is used to control the visibility of control.

Android CheckBox Control Example

Following is the example of defining multiple CheckBox controls and one Button control in LinearLayout to get the selected values of CheckBox controls when we click on Button in the android application.

Create a new android application using android studio and give names as CheckBoxExample.

Now open an activity_main.xml file from \res\layout path and write the code like as shown below

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical" android:layout_width="match_parent"
   android:layout_height="match_parent">
```

```
<CheckBox
    android:id="@+id/chkJava"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginTop="150dp"
    android:layout_marginLeft="100dp"
    android:text="Java"
    android:onClick="onCheckboxClicked"/>
<CheckBox
    android:id="@+id/chkPython"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginLeft="100dp"
    android:text="Python"
    android:onClick="onCheckboxClicked"/>
<CheckBox
    android:id="@+id/chkAndroid"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginLeft="100dp"
    android:text="Android"
    android:onClick="onCheckboxClicked"/>
<CheckBox
    android:id="@+id/chkAngular"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:padding="10dp"
    android:layout_marginLeft="100dp"
    android:text="AngularJS"
    android:onClick="onCheckboxClicked"/>
<Button
    android:id="@+id/getBtn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:layout_marginLeft="100dp"
        android:text="Get Details" />
</LinearLayout>
```

If you observe above code we created a multiple CheckBox controls and one Button control in XML Layout file.

Once we are done with the creation of layout with required controls, we need to load the XML layout resource from our activity onCreate() callback method, for that open main activity file MainActivity.java from \java\com.tutlane.checkboxexample path and write the code like as shown below.

MainActivity.java

```
package com.tutlane.checkboxexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    CheckBox android, java, angular, python;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        android = (CheckBox)findViewById(R.id.chkAndroid);
        angular = (CheckBox)findViewById(R.id.chkAngular);
        java = (CheckBox)findViewById(R.id.chkJava);
        python = (CheckBox)findViewById(R.id.chkPython);
        Button btn = (Button)findViewById(R.id.getBtn);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String result = "Selected Courses";
                if(android.isChecked()){
                result += "\nAndroid";
```

```
            }
            if(angular.isChecked()){
               result += "\nAngularJS";
            }
            if(java.isChecked()){
               result += "\nJava";
            }
            if(python.isChecked()){
               result += "\nPython";
            }
            Toast.makeText(getApplicationContext(), result, Toast.LENGTH_SHORT).show();
         }
      });
   }
   public void onCheckboxClicked(View view) {
      boolean checked = ((CheckBox) view).isChecked();
      String str="";
      // Check which checkbox was clicked
      switch(view.getId()) {
         case R.id.chkAndroid:
            str = checked?"Android Selected":"Android Deselected";
            break;
         case R.id.chkAngular:
            str = checked?"AngularJS Selected":"AngularJS Deselected";
            break;
         case R.id.chkJava:
            str = checked?"Java Selected":"Java Deselected";
            break;
         case R.id.chkPython:
            str = checked?"Python Selected":"Python Deselected";
            break;
      }
      Toast.makeText(getApplicationContext(), str, Toast.LENGTH_SHORT).show();
   }
}
```

If you observe above code we are calling our layout using setContentView method in the form of R.layout.layout_file_name in our activity file. Here our xml file name

is activity_main.xml so we used file name activity_main and we are getting the status of CheckBox controls when they Select / Deselect and getting the selected CheckBox control values on Button click.

Generally, during the launch of our activity, onCreate() callback method will be called by android framework to get the required layout for an activity.

Output of Android CheckBox Example
When we execute the above example using the android virtual device (AVD) we will get a result like as shown below.

If you observe above result, we are able to get the status of checkboxes while selecting / deselecting and getting all the selected CheckBox values on button click.

This is how we can use CheckBox control in android applications to allow users to select one or more options based on our requirements.

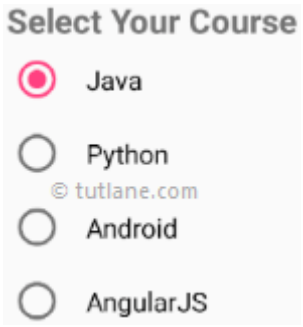
6. RadioButton:

In android, **Radio Button** is a two-states button that can be either checked or unchecked and it's the same as CheckBox control, except that it will allow only one option to select from the group of options.

The user can press or click on the radio button to make it select. In android, CheckBox control allow users to change the state of control either Checked or Unchecked but the radio button cannot be unchecked once it is checked.

Generally, we can use **RadioButton** controls in an android application to allow users to select only one option from the set of values.

Following is the pictorial representation of using **RadioButton** control in android applications.



In android, we use radio buttons with in a **RadioGroup** to combine multiple radio buttons into one group and it will make sure that users can select only one option from the group of multiple options.

By default, the android **RadioButton** will be in **OFF** (**Unchecked**) state. We can change the default state of **RadioButton** by using **android:checked** attribute.

In case, if we want to change the state of **RadioButton** to **ON** (**Checked**), then we need to set **android:checked = "true"** in our XML layout file.

In android, we can create **RadioButton** control in two ways either in the XML layout file or create it in the Activity file programmatically.

Create RadioButton in XML Layout File:
Following is the sample way to define **RadioButton** control using **RadioGroup** in the XML layout file in the android application.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent" android:layout_height="match_parent">
```

```
<RadioGroup
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical">
   <RadioButton
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Java"
      android:checked="true"/>
</RelativeLayout>
```

If you observe above code snippet, here we defined **RadioButton** control and setting RadioButton state **ON** using **android:checked** attribute in xml layout file.

Handle Android RadioButton Click Events:
Generally, whenever the user click on **RadioButton** to Select or Deselect the **RadioButton** object will receives an **on-click** event.

In android, we can define RadioButton click event in two ways either in the XML layout file or create it in Activity file programmatically.

Define RadioButton Click Event in Activity File
In android, we can define **RadioButton** click event programmatically in Activity file rather than XML layout file.

To define **RadioButton** click event programmatically, create **View.OnClickListener** object and assign it to the button by calling **setOnClickListener(View.OnClickListener)** like as shown below.

```
RadioButton rdb = (RadioButton) findViewById(R.id.radiobutton1);
rdb.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      boolean checked = ((RadioButton) v).isChecked();
      // Check which radiobutton was pressed
      if (checked){
         // Do your coding
      }
      else{
```

```
            // Do your coding
        }
    }
});
```

This is how we can handle **RadioButton** click events in android applications based on our requirements.

Android RadioButton Control Attributes:
Following are the some of commonly used attributes related to **RadioButton** control in android applications.

Attribute	Description
android:id	It is used to uniquely identify the control
android:checked	It is used to specify the current state of radio button
android:gravity	It is used to specify how to align the text like left, right, center, top, etc.
android:text	It is used to set the text for the radio button.
android:textColor	It is used to change the color of text.
android:textSize	It is used to specify the size of the text.
android:textStyle	It is used to change the style (bold, italic, bolditalic) of text.
android:background	It is used to set the background color for radio button control.
android:padding	It is used to set the padding from left, right, top and bottom.
android:onClick	It's the name of the method to invoke when the radio button clicked.
android:visibility	It is used to control the visibility of control.

Android RadioButton Control Example

Following is the example of defining a multiple **RadioButton** controls, one TextView control and one Button control in RelativeLayout to get the selected values of RadioButton controls when we click on Button in the android application.

Create a new android application using android studio and give names as **RadioButtonExample**.

Now open an **activity_main.xml** file from **\res\layout** path and write the code like as shown below

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent" android:layout_height="match_parent">
   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="150dp"
      android:layout_marginLeft="100dp"
      android:textSize="18dp"
      android:text="Select Your Course"
      android:textStyle="bold"
      android:id="@+id/txtView"/>
<RadioGroup
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:orientation="vertical"
   android:id="@+id/rdGroup"
   android:layout_below="@+id/txtView">
   <RadioButton
      android:id="@+id/rdbJava"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:padding="10dp"
      android:layout_marginLeft="100dp"
      android:text="Java"
      android:onClick="onRadioButtonClicked"/>
   <RadioButton
      android:id="@+id/rdbPython"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginLeft="100dp"
        android:text="Python"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton
        android:id="@+id/rdbAndroid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginLeft="100dp"
        android:text="Android"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton
        android:id="@+id/rdbAngular"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:layout_marginLeft="100dp"
        android:text="AngularJS"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
    <Button
        android:id="@+id/getBtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_below="@+id/rdGroup"
        android:text="Get Course" />
</RelativeLayout>
```

If you observe above code we created a multiple **RadioButton** controls, one TextView control and one Button control in XML Layout file.

Once we are done with the creation of layout with required controls, we need to load the XML layout resource from our activity **onCreate()** callback method, for that open main activity file **MainActivity.java** from **\java\com.tutlane.radiobuttonexample** path and write the code like as shown below.

MainActivity.java

```
package com.tutlane.radiobuttonexample;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    RadioButton android, java, angular, python;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        android = (RadioButton)findViewById(R.id.rdbAndroid);
        angular = (RadioButton)findViewById(R.id.rdbAngular);
        java = (RadioButton)findViewById(R.id.rdbJava);
        python = (RadioButton)findViewById(R.id.rdbPython);
        Button btn = (Button)findViewById(R.id.getBtn);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String result = "Selected Course: ";
                result+= (android.isChecked())?"Android":(angular.isChecked())?"AngularJS":
(java.isChecked())?"Java":(python.isChecked())?"Python":"";
                Toast.makeText(getApplicationContext(), result, Toast.LENGTH_SHORT).sho
w();
            }
        });
    }
    public void onRadioButtonClicked(View view) {
        boolean checked = ((RadioButton) view).isChecked();
        String str="";
        // Check which radio button was clicked
        switch(view.getId()) {
            case R.id.rdbAndroid:
                if(checked)
                str = "Android Selected";
```

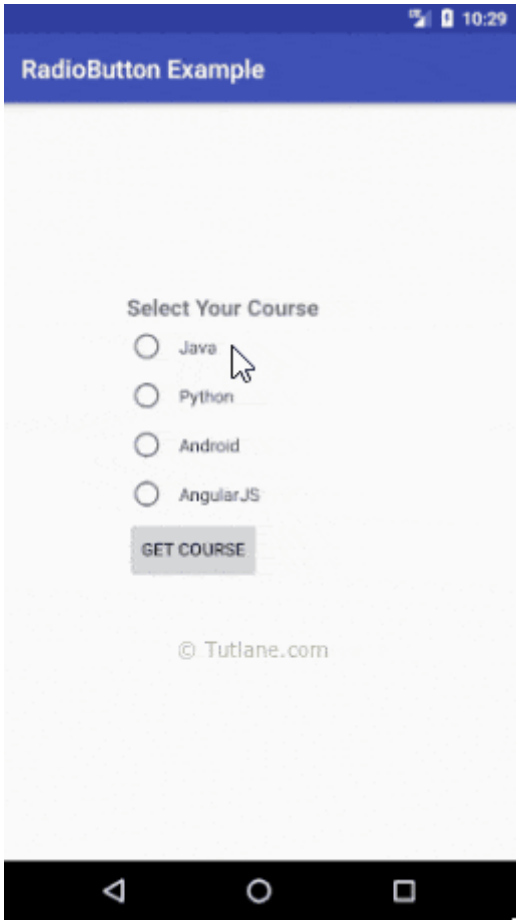
```
            break;
        case R.id.rdbAngular:
            if(checked)
            str = "AngularJS Selected";
            break;
        case R.id.rdbJava:
            if(checked)
            str = "Java Selected";
            break;
        case R.id.rdbPython:
            if(checked)
            str = "Python Selected";
            break;
      }
      Toast.makeText(getApplicationContext(), str, Toast.LENGTH_SHORT).show();
   }
}
```

If you observe above code we are calling our layout using **setContentView** method in the form of **R.layout.layout_file_name** in our activity file. Here our xml file name is **activity_main.xml** so we used file name **activity_main** and we are getting the status of **RadiButton** controls when they **Select** / **Deselect** and getting selected RadioButton control value on Button click.

Generally, during the launch of our activity, **onCreate()** callback method will be called by the android framework to get the required layout for an activity.

Output of Android RadioButton Example

When we run the above example using android virtual device (AVD) we will get a result like as shown below.



If you observe the above result, we are able to select only one option from the set of values and getting the selected **RadioButton** value on button click.

This is how we can use **RadioButton** control in android applications to allow users to select only one option from a set of values based on our requirements.